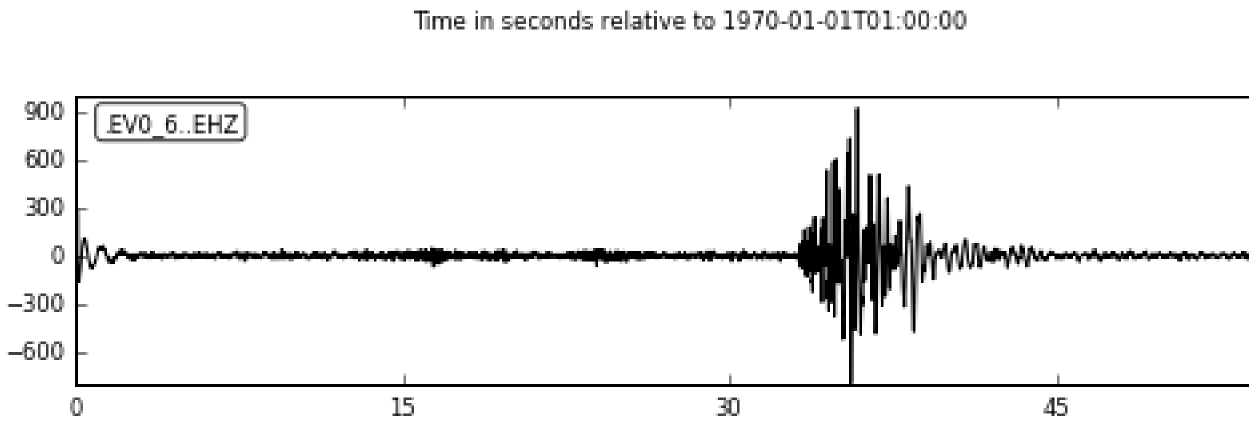
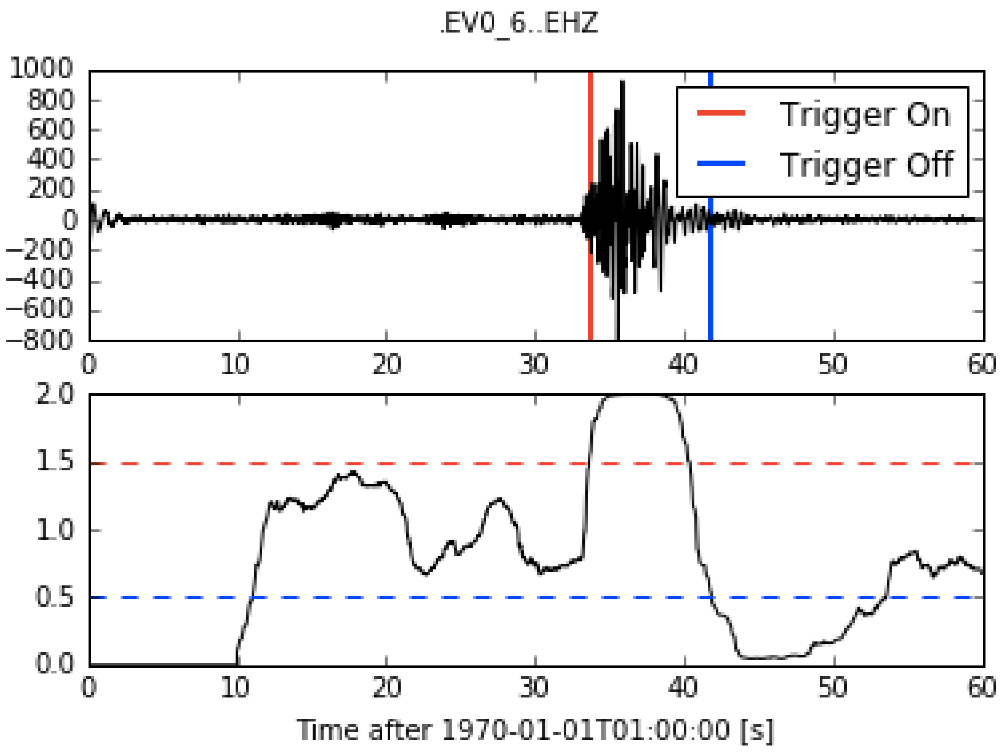
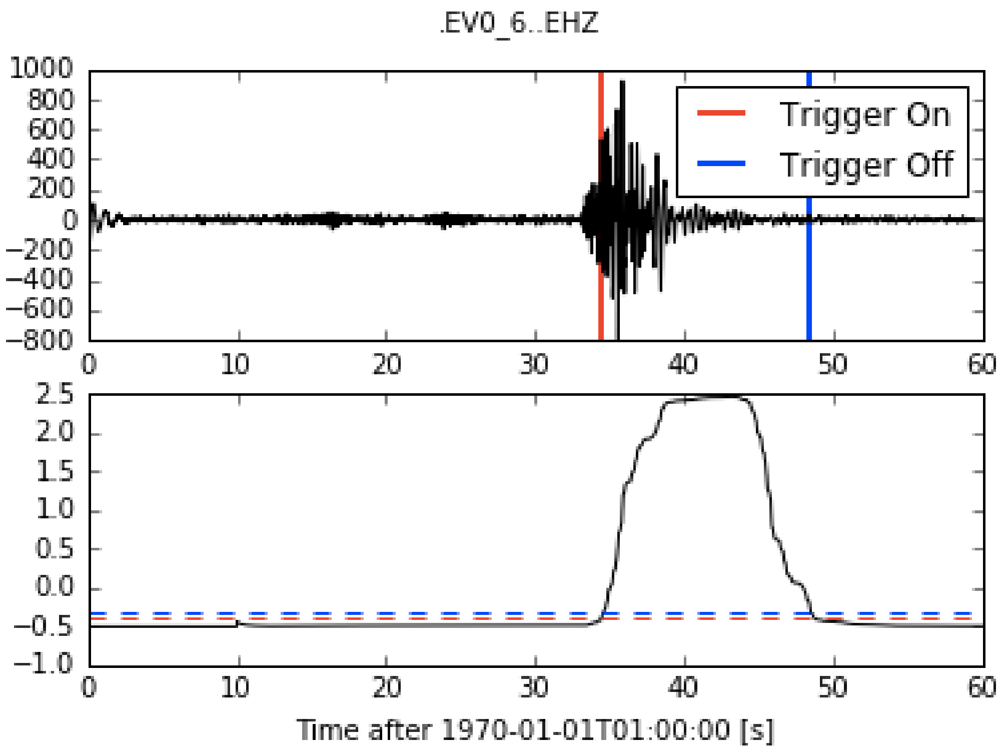
```
In [29]:

In [30]: from obspy.core import read
    ...: st = read("https://examples.obspy.org/ev0_6.a01.gse2")
    ...: st = st.select(component="Z")
    ...: tr = st[0]

In [31]: tr.stats
    ...: df = tr.stats.sampling_rate

In [32]: tr.plot(type="relative")
```



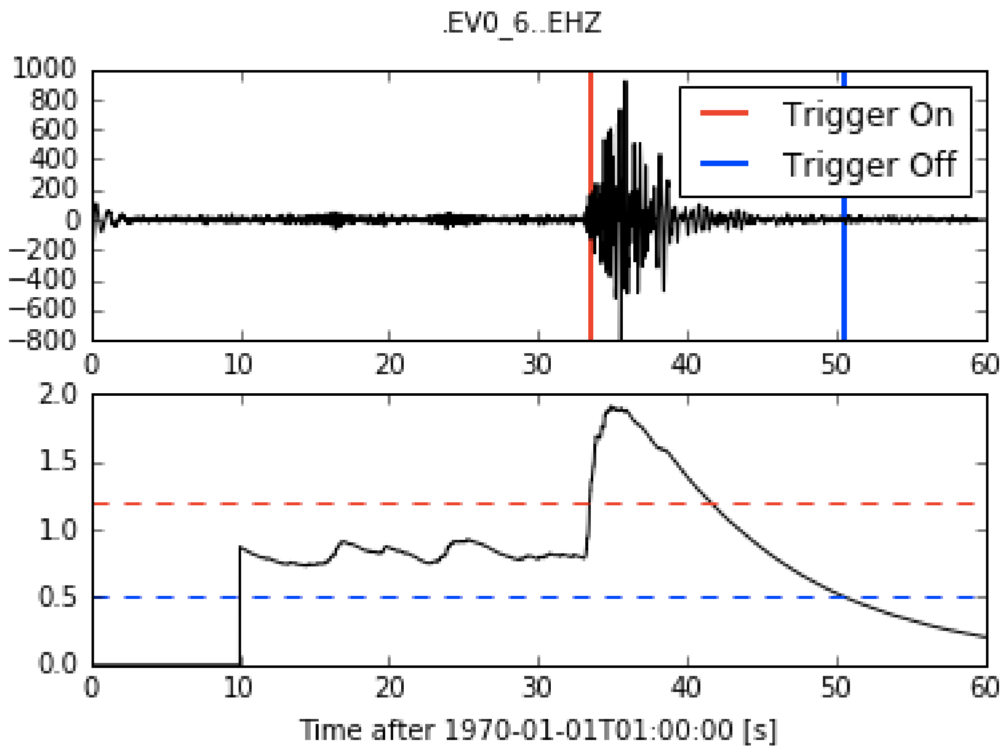
```
In [33]: from obspy.signal.trigger import classic_sta_lta
    ...: help(classic_sta_lta)
    ...: staltaratio = classic_sta_lta(tr.data, int(5*df),
int(10*df))
Help on function classic_sta_lta in module obspy.signal.trigger:

classic_sta_lta(a, nsta, nlta)
    Computes the standard STA/LTA from a given input array a. The
length of
    the STA is given by nsta in samples, respectively is the
length of the
    LTA given by nlta in samples.

    Fast version written in C.

    :type a: NumPy :class:`~numpy.ndarray`
    :param a: Seismic Trace
    :type nsta: int
    :param nsta: Length of short time average window in samples
    :type nlta: int
    :param nlta: Length of long time average window in samples
    :rtype: NumPy :class:`~numpy.ndarray`
    :return: Characteristic function of classic STA/LTA


In [34]: from obspy.signal.trigger import plot_trigger
    ...: help(plot_trigger)
    ...: plot_trigger(tr, staltaratio, 1.5, 0.5)
Help on function plot_trigger in module obspy.signal.trigger:

plot_trigger(trace, cft, thr_on, thr_off, show=True)
    Plot characteristic function of trigger along with waveform
data and
    trigger On/Off from given thresholds.

    :type trace: :class:`~obspy.core.trace.Trace`
    :param trace: waveform data
    :type cft: :class:`numpy.ndarray`
    :param cft: characteristic function as returned by a trigger
in
        :mod:`obspy.signal.trigger`
    :type thr_on: float
    :param thr_on: threshold for switching trigger on
    :type thr_off: float
    :param thr_off: threshold for switching trigger off
    :type show: bool
    :param show: Do not call `plt.show()` at end of routine. That
way,
        further modifications can be done to the figure before
showing it.
```



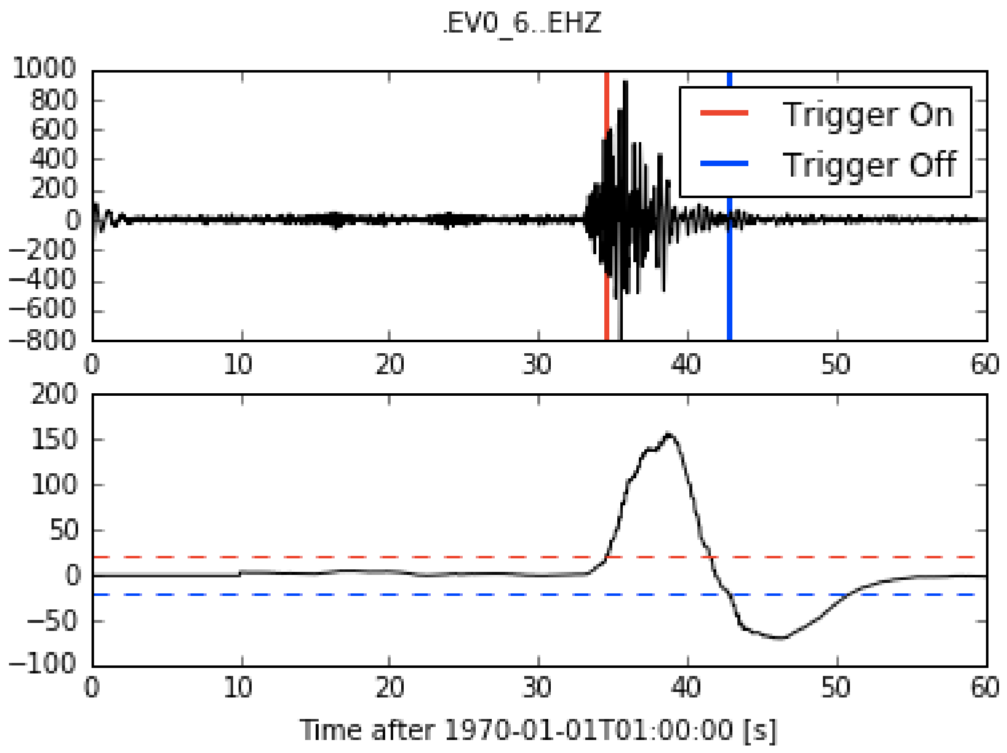
```
In [35]: from obspy.signal.trigger import z_detect
    ...: help(z_detect)
    ...: staltaratio = z_detect(tr.data, int(df*10))
    ...: plot_trigger(tr, staltaratio, -0.4, -0.3)
Help on function z_detect in module obspy.signal.trigger:

z_detect(a, nsta)
    Z-detector.

    :param nsta: Window length in Samples.

    .. seealso:: [Withers1998]_, p. 99
```



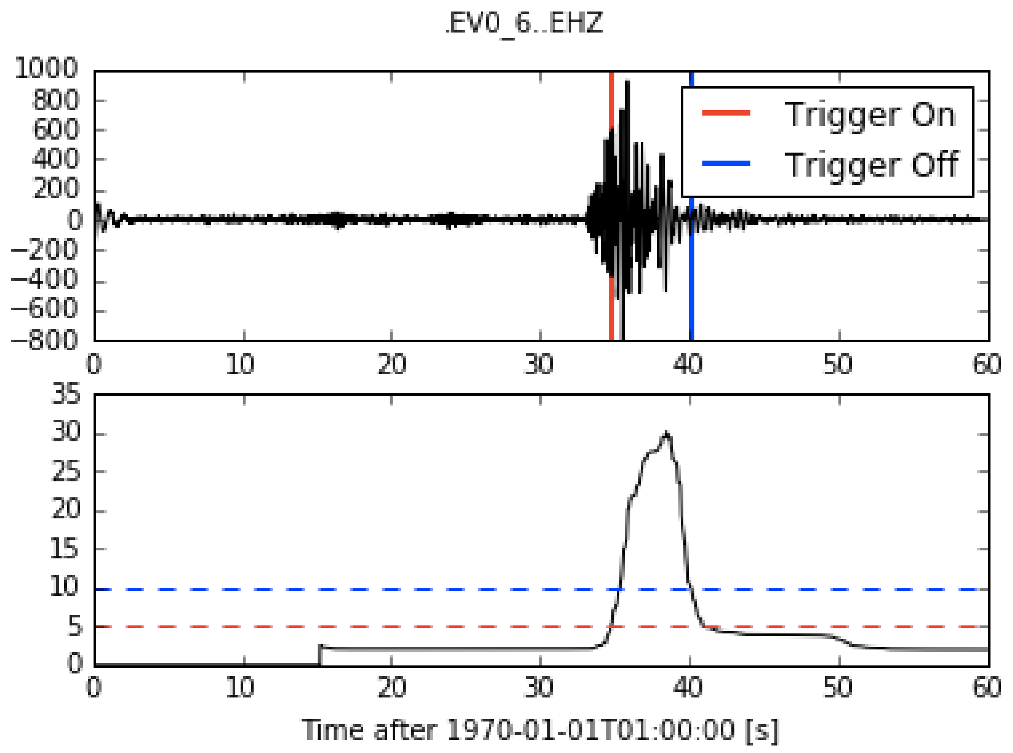
```
In [36]: from obspy.signal.trigger import recursive_sta_lta
    ...: staltaratio = recursive_sta_lta(tr.data, int(5 * df),
int(10 * df))
    ...: help('recursive_sta_lta')
    ...: plot_trigger(tr, staltaratio, 1.2, 0.5)
no Python documentation found for 'recursive_sta_lta'
```



```
In [37]: from obspy.signal.trigger import carl_sta_trig
    ...: help('carl_sta_trig')
    ...: staltaratio = carl_sta_trig(tr.data, int(df * 5), int(10
* df), 0.8, 0.8)
    ...: plot_trigger(tr, staltaratio, 20.0, -20.0)
no Python documentation found for 'carl_sta_trig'
```

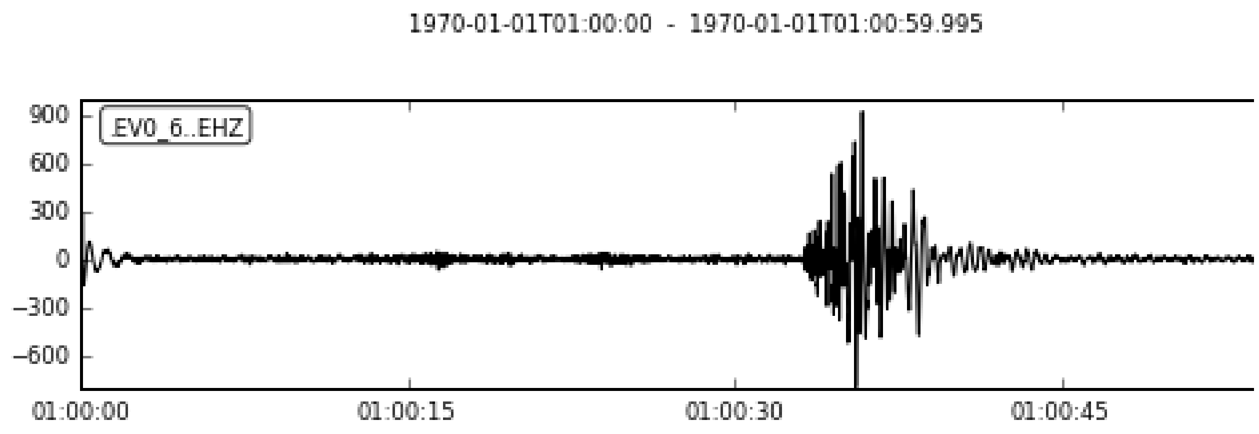


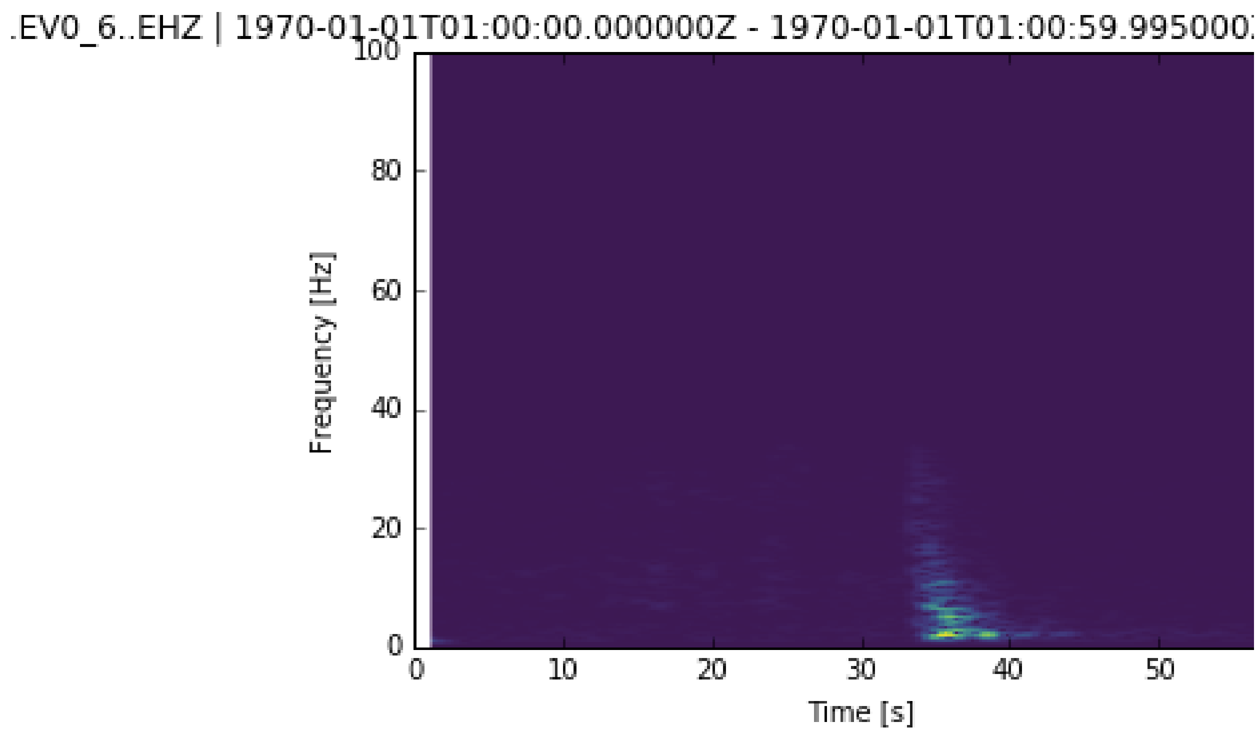
```
In [38]: from obspy.signal.trigger import delayed_sta_lta
    ...: staltaratio = delayed_sta_lta(tr.data, int(5*df),
int(10*df))
    ...: plot_trigger(tr, staltaratio, 5, 10)
```



```
In [39]: import sys
    ...: 
sys.path.append('/Users/glennthompson/Dropbox/scratch_matlab')
    ...: import tune_sta_lta as tsl
    ...: help(tsl.tune_sta_lta)
Help on function tune_sta_lta in module tune_sta_lta:

tune_sta_lta(tr, algorithm, tsignalstart, tsignalend, ntrys)
    tune_sta_lta Tune (optimize) STA and LTA window lengths for 
different
        STA/LTA algorithms to maximise the characteristic STA:LTA 
function

        Inputs:
            tr - a trace object
            algorithm - the STA/LTA method to use from
                   ['classic_sta_lta', 'z_detect', 
'recursive_sta_lta', 'carl_sta_trig', 'delayed_sta_lta']
            tsignalstart, tsignalend - the number of seconds into 
the trace object where the target signal lies
            thresh_on, thresh_off - used in plot_trigger plot 
only
            ntrys = number of STA/LTA window combinations to try
        Outputs:
            a list (called result) containing:
                sta_best - length of best STA window (seconds)
                lta_best - length of best LTA window (seconds)
                staltaratio_best - the characteristic function
returned for these windows

        To do:
            Support filtering?

    Example:

    # import the tune_sta_lta package
    import sys
    sys.path.append('/path/to/directory/containing/tune_sta_lta.p
y')
    import tune_sta_lta as tsl

    # read a seismogram into a trace object
    from obspy.core import read
    st = read("https://examples.obspy.org/ev0_6.a01.gse2")
    st = st.select(component="Z")
    tr = st[0]

    # call the tune_sta_lta function
    algorithm - 'classic_sta_lta'
    tsignalstart = 30.0
    tsignalend = 40.0
    ntrys = 100

    tsl.tune_sta_lta(tr, algorithm, tsignalstart, tsignalend,
ntrys)
```

```
In [40]: from obspy.core import read
    ...: st = read("https://examples.obspy.org/ev0_6.a01.gse2")
    ...: st = st.select(component="Z")
    ...: tr = st[0]

In [41]: tr.plot()
    ...:
    ...: # plot spectrogram
    ...: tr.spectrogram()
```





```
In [42]: algorithm = 'classic_sta_lta'
    ...: TSIGNAL_START = 30.0
    ...: TSIGNAL_END = 40.0
    ...: NTRIES=100
    ...: tsl.tune_sta_lta(tr, algorithm, TSIGNAL_START,
TSIGNAL_END, NTRIES)

Algorithm: classic_sta_lta
sta_seconds=1.5 lta_seconds=15.0 max(staltaratio)=9.6
Out[42]:
[1.5, 15.0, array([ 0.        ,  0.        ,  0.        , ...,
0.36682018,
         0.36657746,  0.36660746])]
```

```
In [43]:
```